Developmental Simulations with Cellerator

Bruce E. Shapiro
Jet Propulsion Laboratory
M/S 126-347, 4800 Oak Grove Drive
Pasadena, CA 91109
bshapiro@jpl.nasa.gov

Eric D. Mjolsness
Jet Propulsion Laboratory
M/S 126-347, 4800 Oak Grove Drive
Pasadena, CA 91109
Eric.D.Mjolsness@jpl.nasa.gov

ABSTRACT

In this paper, we describe how Cellerator can be used to perform developmental simulations. Biochemical reactions are specified in Cellerator using a compact arrow-based notation, which are then automatically translated into the appropriate ordinary differential equations. These reactions can be combined into modules, leading to a natural graph-based hierarchical implementation. We demonstrate how the paradigm of organisms-as-graphs can represent the basic features of developing tissue, and propose a variable-structure graph-based algorithm to describe simple developmental processes. In particular, we show how such a variable-structure system (VSS) can be implemented using a pre-packaged fixed-structure differential equation solver.

1. INTRODUCTION

Cellerator is a Mathematica package designed to facilitate biological modeling via automated equation generation [10]. The implementation is based on the concept of a hierarchy of *canonical forms* that describe biological processes at various levels of detail. At each level of hierarchy there are two classes of canonical forms: the *input canonical form*, which is used to supply information to the program, and the *output canonical form*, which is automatically generated by the simulator.

To understand canonical forms, consider the usual method of describing biological systems in terms of signal transduction networks (STN). Such networks are by their very nature hierarchical; the individual nodes of an STN may represent anything from single molecules (e.g., a particular enzyme or receptor molecule), through simple modules composed of a set of ubiquitous processes (e.g., a MAPK cascade or a particular transcription complex), to extremely complex STNs. The links between the nodes represent their (mutual) interactions including input and output signals. At the highest level of abstraction, a cartoon description of the network itself would be the input canonical form, and a complete set of differential equations describing the network would be the corresponding output canonical form.

The Cellerator paradigm is based on the proposition that there is a one-to-one relationship between each class of interaction and/or node in an STN and a formal (e.g., mathematical) description of that process. An input canonical form for an enzymatic reaction, for example, might be written as $E(S \Rightarrow P)$ meaning that enzyme E facilitates the conversion of S (the "source") into P (the product). One corresponding output canonical form would be the set of differential equations determined by the law of mass action. A slightly different arrow notation, such as $E(S \rightarrow P)$ could instead indicate that the desired output canonical form should be the steady state (Michaelis-Menten) equations. Cellerator represents STN nodes by *variables* (e.g., concentration of a chemical species) and links by *arrows* (e.g., for enzymatic reactions).

A wide variety of different arrow notations have been implemented in Cellerator to describe the input canonical forms for biological systems. The output canonical forms produced by Cellerator take two forms: reaction networks (standard biochemical equations), and systems of ordinary differential equations (ODEs) for the concentrations of chemical species. While no cartoon-based GUI interface has yet been implemented, the Cellerator paradigm allows for the addition of a graphical front-end that could produce the necessary input canonical forms.

The Cellerator implementation allows explicit output description at each level so that "power-users" can modify the equations at any stage desired. They can thus manually modify the equations or add additional constraints in the form of differential, algebraic, or chemical equations. The output canonical forms are produced in a variety of formats: as Mathematica differential equations, in C, FORTRAN, SBML [5], MATHML, or HTML. If desired, the user can also solve the equations numerically (using Mathematica's NDSolve [11]).

Multi-cellular developmental processes have (at least) one additional layer of complexity. In this case it is possible to represent a tissue, such as the shoot meristem of a plant, as a mega-STN where the nodes represent cells and the links represent inter-cellular interactions. At many stages of development it should be possible to describe the essential processes with only a small number of cells, so the number of nodes does not present a computational obstacle. Complexity arises from the interactions. There may be many different signal transduction networks that need to be represented within any given cell, each of which may be mutually interacting, both intra- and inter-cellularly. In particular there will probably be many instantiations of essentially the same network (e.g., mitotic oscillators) in each cell. Even worse, there could be multiple instantiations of the same network (e.g. MAP-Kinase cascades or transcription complex formation) within the same cell. An even more difficult issue is how to deal with birth and death processes. Since either the birth or death of a cell would cause the total number of cells to change during a simulation, the corresponding number of differential equations necessary to describe the overall STN changes with time: we have a variable structure system.

We postulate that the paradigm of organisms-as-graphs can represent many of the basic features of developing tissue, and propose a variable-structure graph-based algorithm to describe simple developmental processes in this paper. In particular, we show how such a variable-structure system can be implemented using a pre-packaged fixed-structure differential equation solver. We then proceed to show how this has been done in Cellerator, and finally illustrate the overall process with the design of a minimal system for developmental simulation.

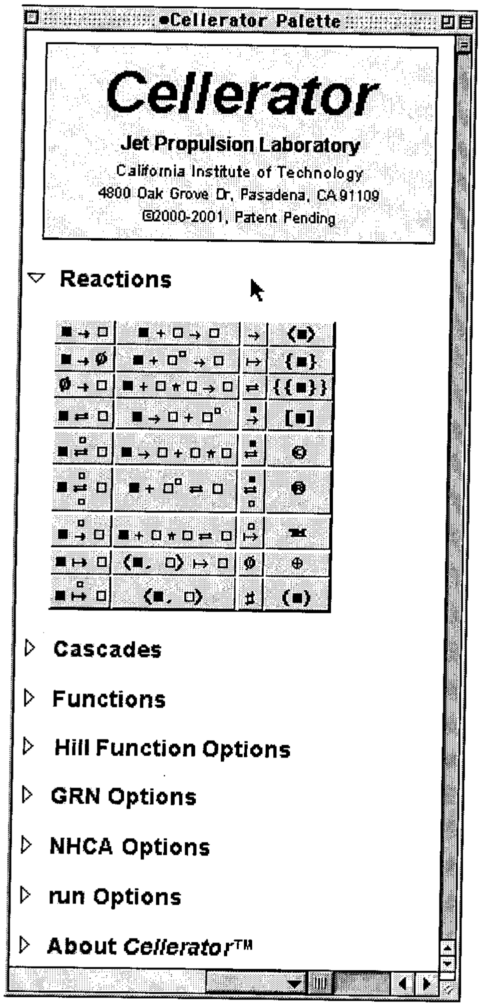


Figure 1. Cellerator Palette.

2. METHODS

2.1 Cellerator Canonical Forms

The Cellerator input canonical forms are expressed via a specialized palette selectable arrow notation (Figure 1). As will all Mathematica palettes, "power users" can rearrange the palette as desired.

To illustrate Cellerator notation, consider the glycolytic step in which an activated form of the enzyme phospofructokinase (PFKA) catalyzes the phosphorylation of fructose 6-phosphate (F6P), converting ATP to ADP in the process,

$$F6P + PFKA + ATP \Leftrightarrow X \rightarrow F6P^{*} + PFKA + ADP \qquad (1)$$

where X is the sequence of intermediate compounds that are formed during the process. In a reduced model of glycolysis [4], PFK catalyzes the removal of a phosphate from ATP,

$$ATP + PFKA \underset{k_{-1}}{\overset{k_1}{\rightleftharpoons}} Y \overset{k_2}{\rightarrow} ADP + PFKA \qquad (2)$$

where Y is the intermediate compound formed by PFKA and ATP. The Cellerator input canonical form for (2) is

$$ATP \overset{PFKA}{\rightleftharpoons} ADP \qquad (3)$$

To specify the rate constants shown in (2), we would express the reaction as the Mathematica list

$$\{ATP \overset{PFKA}{\rightleftharpoons} ADP, k_1, k_{-1}, k_2\} \qquad (4)$$

Unspecified rate constants take on default values. In addition to equation (2) ADP activates PFK, ATP is continually produced, and ADP is continually degraded:

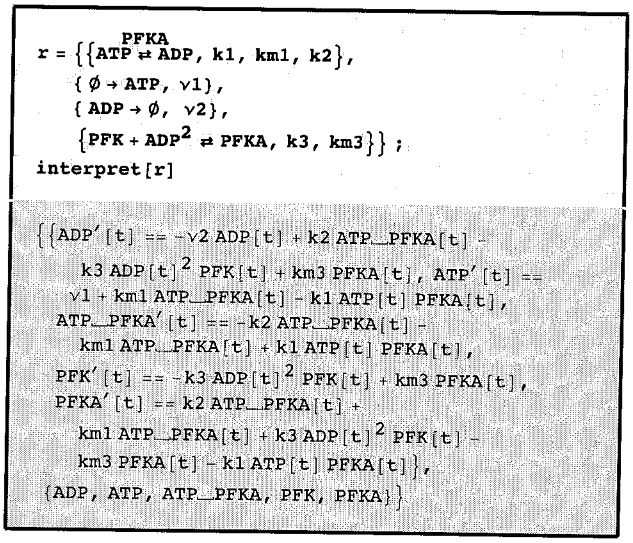
$$\overset{v_1}{\rightarrow} ATP \qquad (5)$$

$$ADP \overset{v_2}{\rightarrow} \qquad (6)$$

$$2ADP + PFK \underset{k_{-3}}{\overset{k_3}{\rightleftharpoons}} PFKS \qquad (7)$$

The canonical form for conversion of A to B is $A \rightarrow B$. To describe (5) and (6), Cellerator uses the special symbol $\varnothing$ to indicate conversion to (annihilation) or from (creation) the empty set. A bi-directional arrow ($\rightleftharpoons$) indicates that the reaction can proceed in both ways, as in equation (7).

The reactions, once expressed in canonical form, can then be translated to differential equations according to the law of mass action via the *interpret* function. The complete set of canonical forms for the simplified glycolytic model, along with the corresponding set of automatically generated differential equations, is illustrated in Figure 2. When the same chemical species X occurs in multiple reactions, one term (or set of terms) is generated in the differential equation for $X'(t)$ for each reaction. The final interpreted differential equation then gives the sum of all these reaction terms (see, for example, the differential equation for ADP in figure 2). Some other Cellerator canonical forms are discussed in the following paragraphs.



```
         PFKA
r = {{ATP ⇄ ADP, k1, km1, k2},
    { ∅ → ATP, v1},
    { ADP → ∅, v2},
    {PFK + ADP^2 ⇄ PFKA, k3, km3}} ;
interpret[r]
```

```
{{ADP′[t] == -v2 ADP[t] + k2 ATP⎵PFKA[t] -
     k3 ADP[t]^2 PFK[t] + km3 PFKA[t], ATP′[t] ==
    v1 + km1 ATP⎵PFKA[t] - k1 ATP[t] PFKA[t],
  ATP⎵PFKA′[t] == -k2 ATP⎵PFKA[t] -
     km1 ATP⎵PFKA[t] + k1 ATP[t] PFKA[t],
  PFK′[t] == -k3 ADP[t]^2 PFK[t] + km3 PFKA[t],
  PFKA′[t] == k2 ATP⎵PFKA[t] +
     km1 ATP⎵PFKA[t] + k3 ADP[t]^2 PFK[t] -
     km3 PFKA[t] - k1 ATP[t] PFKA[t]},
 {ADP, ATP, ATP⎵PFKA, PFK, PFKA}}
```

Figure 2. Canonical form for glycolysis reactions and corresponding ODEs generated by Cellerator.

2.1.1 *Uncatalyzed mass action reactions*

The general Cellerator uncatalyzed mass action reaction is

$$A+B^n \rightarrow C \tag{8}$$

where B is optional and n is an optional positive integer indicating that one molecule of A combines with n molecules of B form one molecule of C. In the absence of B, either A or C may be the empty set ($\varnothing$) to indicate creation or annihilation. The term B^n can also be written as nB. Two-way reactions are written with the "double arrow" as

$$A+B^n \rightleftharpoons C \tag{9}$$

The general syntax for generating ODEs from reactions is

$$\texttt{interpret[r]} \tag{10}$$

where r is a list of reactions of the form

$$\begin{aligned}&\texttt{r=\{\{reaction,rates\},}\\ &\qquad\texttt{\{reaction,rates\},...\}}\end{aligned} \tag{11}$$

Several examples of the translation are illustrated in table 1.

Table 1. Cellerator arrows for uncatalyzed reactions.

Reaction Syntax	ODE Interpretation
$\{S \rightarrow P,k\}$	$S' = -P' = -kS$
$\{A+B \rightarrow C,k\}$	$A' = B' = -C' = -kAB$
$\{A+B^n \rightarrow C,k\}$	$A' = B' = -C' = -kAB^n$
$\{A \rightleftharpoons B,kf,kr\}$	$A' = -B' = -k_f A + k_r B$
$\{A+B \rightleftharpoons C,kf,kr\}$	$A' = B' = -C' = -k_f AB + k_r C$
$\{\varnothing \rightarrow A,k\}$	$A' = k$
$\{B \rightarrow \varnothing,k\}$	$B' = -kB$

2.1.2 *Catalytic reactions*

By a *catalytic* reaction we mean any reaction in which one molecule E (for "enzyme") catalyzes the conversion S (for "source") into P (for "product"). In the most generalized form, an intermediate species (called SE) is formed,

$$S+E \underset{d}{\overset{a}{\rightleftharpoons}} SE \xrightarrow{k} P+E \tag{12}$$

where a, d, and k are rate constants. The conversion of ATP into ADP by PFKA in equation (4) is an example of such a reaction. The canonical form for a catalytic reaction is

$$\{S \overset{E}{\rightleftharpoons} P,a,d,k\} \tag{13}$$

which is translated into the canonical forms

$$\{\{S+E \rightarrow SE,a\},\{SE \rightarrow S+E,d\},\{SE \rightarrow P+E,k\}\} \tag{14}$$

and interpreted as already described. If a second enzyme F catalyzes the reverse reaction

$$P+F \underset{d_1}{\overset{a_1}{\rightleftharpoons}} PF \xrightarrow{k_1} S+F \tag{15}$$

we can either add a second reaction

$$\{P \overset{F}{\rightleftharpoons} S,a1,d1,k1\} \tag{16}$$

or we can use the different notation

$$\{S \underset{F}{\overset{E}{\rightleftharpoons}} P,a,d,k,a1,d1,k1\} \tag{17}$$

to indicate all six reactions described by equations (13) and (16). If no intermediate compound is formed, we can write

$$\{S \overset{E}{\rightarrow} P,k\} \tag{18}$$

A summary of catalytic reactions is given in table 2.

Table 2. Cellerator arrows for catalyzed reactions.

Reaction Syntax	ODE interpretation
$\{S \overset{E}{\rightleftharpoons} P,a,d,k\}$	$S' = -a \cdot E \cdot S + d \cdot S$ $P' = k \cdot (SE)$ $E' = -a \cdot E \cdot S + (d+k) \cdot (SE) = -(SE)'$
$\{S \underset{F}{\overset{E}{\rightleftharpoons}} P,a,d,k,$ $a1,d1,k1\}$	$S' = k_1 \cdot (PF) - a \cdot E \cdot S + d \cdot (SE)$ $P' = -a_1 \cdot F \cdot P + d_1 \cdot (PF) + k \cdot (SE)$ $E' = -a \cdot E \cdot S + (d+k) \cdot (SE) = -(SE)'$ $F' = -a_1 \cdot F \cdot P + (d_1 + k_1) \cdot (PF) = -(PF)'$
$\{S \overset{E}{\rightarrow} P,k\}$	$S' = -k \cdot E \cdot S = -P'$
$\{S \mapsto P\}$	$S' = \dfrac{(k+vE)S^n}{K^n + S^n} = -P'$

2.1.3 *Transcriptional regulation*

Several forms of transcriptional regulation are provided with the left-bar arrow (which is also used to provide hill functions). The input canonical forms is

$$\{A \mapsto B,\ f[\text{options}]\} \tag{19}$$

where f is a function that indicates the format of the regulatory function, and "options" is a list of optional rules that define system parameters (constants). Regulatory functions (values of "f") currently available are: hill (Hill functions); GRN (for *Genetic Regulatory Network*) neural-network dynamics; and NHCA (for *Non-Hierarchical Cooperative Activation* [8]) a form of pseudo-Monod-Wyman-Changeux dynamics [10]. These are described in the following subsections.

2.1.3.1 *Transcription regulation by hill function*

A reaction of the form

$$\begin{aligned}&\{A \mapsto B,\texttt{hill[vmax} \rightarrow v,\texttt{nhill} \rightarrow n,\\ &\qquad\qquad \texttt{khalf} \rightarrow K,\texttt{basalRate} \rightarrow \{r_0,r_1\}\}\end{aligned} \tag{20}$$

is interpreted as the differential equation

$$B' = r_0 + \frac{r_1 + vA^n}{K^n + A^n} \tag{21}$$

The concentration of species A is not affected by the reaction. If a set of p reactions of the form

$$\{\{\mathtt{A}_1 \mapsto \mathtt{B,hill}[\cdots]\},\ldots,\{\mathtt{A}_\mathtt{p} \mapsto \mathtt{B,hill}[\cdots]\}\} \tag{22}$$

where the hill options have been suppressed for clarity, the differential equation becomes

$$B' = r_0 + \frac{(r_1 + \sum_{i=1}^{p} v_i A_i)^n}{K^n + (r_1 + \sum_{i=1}^{p} v_i A_i)^n} \tag{23}$$

The set of parameters v_i are the connection strengths for the corresponding neural network.

2.1.3.2 Genetic Regulatory Networks

A reaction of the form

$$\begin{aligned}\{\mathtt{A} \mapsto \mathtt{B,GRN[RGRN} \to \mathtt{R,TGRN} \to \mathtt{T},\\ \mathtt{nGRN} \to \mathtt{n,hGRN} \to \mathtt{h}\}\end{aligned} \tag{24}$$

is interpreted as the differential equation

$$B' = \frac{R}{1 + e^{-TA^n + h}} \tag{25}$$

Here T is the connection strength and h is the activation threshold. The concentration of species A is not affected by the reaction. If we have a set of p reactions of the form

$$\{\{\mathtt{A}_1 \mapsto \mathtt{B,GRN}[\cdots]\},\ldots,\{\mathtt{A}_\mathtt{p} \mapsto \mathtt{B,GRN}[\cdots]\}\} \tag{26}$$

the differential equation becomes

$$B' = R\left[1 + \exp\left(-\sum_{i=1}^{p} T_i A_i^{n_i} + h_i\right)\right]^{-1} \tag{27}$$

where the T_i and the h_i are the connection strengths and thresholds for activation of B by A_i.

2.1.3.3 Non-hierarchical Cooperative Activation

This type of regulation provides a non-hierarchical reduction of the HCA (hierarchical cooperative activation, [8]) algorithm that has been previously proposed. A reaction of the form

$$\begin{aligned}\{\mathtt{A} \mapsto \mathtt{B,NHCA[TPLUS} \to \mathtt{T}^+\mathtt{,TMINUS} \to \mathtt{T}^-,\\ \mathtt{nNHCA} \to \mathtt{n,kNHCA} \to \mathtt{k},\\ \mathtt{mNHCA} \to \mathtt{M]}\}\end{aligned} \tag{28}$$

is interpreted[1] as the differential equation

$$B' = \frac{(1 + T^+ A^n)^m}{k(1 + T^- A^n)^m + (1 + T^+ A^n)^m} \tag{29}$$

where the T^+, T^- are the connection strengths for activation and inhibition. Alternatively, the option $\mathtt{TNHCA} \to \mathtt{T}$ supercedes the TPLUS and TMINUS options,

$$B' = \frac{(1 + T\Theta(T)A^n)^m}{k(1 - T\Theta(-T)A^n)^m + (1 + T\Theta(T)A^n)^m} \tag{30}$$

where $\Theta(x)$ is the Heaviside step function (see equation (73), below). With a set of p reactions of the form

$$\{\{\mathtt{A}_1 \mapsto \mathtt{B,NHCA}[\cdots]\},\ldots,\{\mathtt{A}_\mathtt{p} \mapsto \mathtt{B,NHCA}[\cdots]\}\} \tag{31}$$

the differential equations (29) and (30) become

$$B' = \frac{\prod_{i=1}^{p}(1 + T_i^+ A_i^{n_i})^m}{k\prod_{i=1}^{p}(1 + T_i^- A_i^{n_i})^m + \prod_{i=1}^{p}(1 + T_i^+ A_i^{n_i})^m} \tag{32}$$

and

$$B' = \frac{\prod_{i=1}^{p}(1 + T_i\Theta(T_i)A_i^{n_i})^m}{k\prod_{i=1}^{p}(1 - T_i\Theta(-T_i)A_i^{n_i})^m + \prod_{i=1}^{p}(1 + T_i\Theta(T_i)A_i^{n_i})^m} \tag{33}$$

respectively. If cooperative binding is allowed the syntax

$$\begin{aligned}\{\langle \mathtt{A}_1,\mathtt{A}_2,\ldots,\mathtt{A}_\mathtt{p}\rangle \mapsto \mathtt{B},\\ \mathtt{NCHA[TPLUS} \to \{\mathtt{T}_1^+,\mathtt{T}_2^+,\ldots\},\ldots]\}\end{aligned} \tag{34}$$

is used, where the NHCA options are specified as lists but are otherwise identical to (28). The corresponding differential equations are

$$B' = \frac{1 + (\sum_{i=1}^{p} T_i^+ A_i^{n_i})^m}{k(\sum_{i=1}^{p} T_i^- A_i^{n_i})^m + (\sum_{i=1}^{p} T_i^+ A_i^{n_i})^m} \tag{35}$$

and

$$B' = \frac{1 + (\sum_{i=1}^{p} T_i\Theta(T_i)A_i^{n_i})^m}{k(\sum_{i=1}^{p}(-T_i)\Theta(-T_i)A_i^{n_i})^m + (\sum_{i=1}^{p} T_i\Theta(T_i)A_i^{n_i})^m} \tag{36}$$

respectively.

2.1.4 Mass conservation

Because Cellerator automatically generates all of the terms in all of the necessary differential equations no additional constraints are needed to force mass concentration. However, this will sometimes lead to the generation of redundant reactions, which the user may want to suppress. For example, suppose that A can exist in either of two forms, A and A^*, where the total mass of $A + A^* = A_T$ is a constant, but that only the activated form A^* can be converted into B. Biochemically we could submit the reactions

$$A \overset{k1}{\to} A^*,\ A^* \overset{k2}{\to} B,\ A + A^* = A_T \tag{37}$$

to Cellerator as:

$$\mathtt{interpret[\{\{A \to AS,k1\},\{AS \to B,k2\}\}]} \tag{38}$$

The interpret function would return the following differential equations:

$$\begin{aligned}&\mathtt{\{A'[t]==-k1\ A[t],}\\ &\mathtt{AS'[t]==k1\ A[t]\ -\ k2\ AS[t],}\\ &\mathtt{B'[t]==k2\ AS[t]\}}\end{aligned} \tag{39}$$

However, if we are not interested in the specific concentration of AS, for example if it is not used elsewhere in the reaction schema, then we can use the "complement" notation

$$\mathtt{interpret[\{\{Comp[A,AT] \to B,k2\}\}]} \tag{40}$$

[1] The plus (+) and minus (-) superscripts are actually not allowed at the present time by Cellerator; the user would have to specify variable names such as TP or TM (or values) instead. Superscripts are used here to clarify the notation. Elsewhere in this paper the same caution should be applied to the (*) superscript and the various subscripts used on variables. The subscripts would actually be indicated with a square bracket notation, e.g. $A_3[\mathrm{t}]$ would be written as A[3][t].

which produces the single differential equation

$$\texttt{B}'[\texttt{t}] == \texttt{k2}\,(\texttt{AT} - \texttt{A}[\texttt{t}]) \qquad (41)$$

The complement notation can be used in any Cellerator reaction; the default total concentration is 1.

2.2 Domains and Fields

To provide an object-oriented representation of organisms-as-graphs we have developed the concepts of *domain* and *field*. A *domain*, in analogy with the standard use of the term in mathematics, is an object-oriented representation of the mathematical domains that are relevant to solving a particular scientific problem. A *field* is a representation of a function that maps domains to the real numbers. We *instantiate* a domain by applying the *expand* function to it.

For example, we can define the integer domains $I(n)$, $I(m,n)$, and $I(m,n,p)$ to represent the sets $\{1,2,\ldots,n\}$, $\{m,m+1,\ldots,n\}$, and $\{m,m+p,m+2p,\ldots,m+qp\}$, where q is the largest integer such that $m+qp \le n$, respectively. Then the expand function, which we will denote by $\mathcal{E}$ in this paper, maps the object representation of the domain to an implementation of that domain, e.g.,

$$\begin{aligned} &\mathcal{E}[I(n)] = \{1,2,\ldots,n\} \\ &\mathcal{E}[I(m,n)] = \{m,m+1,m+2,\ldots,n\} \\ &\mathcal{E}[I(m,n,p)] = \{m,m+p,m+2p,\ldots,m+qp\} \end{aligned} \qquad (42)$$

Thus the expand function applies the implementation; in actual practice we would generally never have to deal with the expanded function itself. To see what a field is, suppose g is any function that operates on the integers. Then the field operator, which we will denote by $\mathcal{F}$ in this paper,

$$\mathcal{F}: D \to D \times \mathcal{R} \qquad (43)$$

associated with any function $f(x): D \to \mathcal{R}$ (here $\mathcal{R}$ denotes the real numbers) generates a set of ordered pairs

$$\mathcal{F}(f,D) = \{(d_i, f(d_i)) \mid d_i \in D\} \qquad (44)$$

We will use the shorthand $f(D)$ for the set

$$f(D) = \{f_i \mid (d_i, f_i) \in \mathcal{F}(f,D),\ d_i \in D\} = \{f(d_i) \mid d_i \in D\} \qquad (45)$$

which we will call the *range of the field* corresponding to *f*.

To illustrate the concept of fields, suppose that $D = I(5,19,3)$ and that $f(x) = x^2$. Then

$$\mathcal{E}[D] = \{5,8,11,14,17\} \qquad (46)$$

and

$$\mathcal{F}[f,D] = \{(5,25),(8,64),(11,111),(14,196),(17,289)\} \qquad (47)$$

$$f[D] = \{25,64,111,196,289\} \qquad (48)$$

Using fields one can also define composite functions that map domains to the real numbers. For example, let $A \subset \mathcal{R}$ and $f: \mathcal{R} \to \mathcal{R}$ be a function. Then we define the sum function $\Sigma(f,A): S[\mathcal{R}] \to \mathcal{R}$ (where $S[U]$ is the subset operator that gives the set of all subsets of the set U)

$$\Sigma(f,A) = \sum\nolimits_{x \in A} f(x) \qquad (49)$$

The sum function applies f to every element of A and then sums the result; equation (49) might be read as "the sum of $f(x)$ over the set A." If D is a domain and we let $A = \mathcal{E}(D)$ we can define the sum operator – which gives the sum of the function $f(x)$ over a domain D – as

$$\sum\nolimits_D f(x) \equiv \Sigma(f, \mathcal{E}[D]) = \sum\nolimits_{x \in \mathcal{E}(D)} f(x) = \sum\nolimits_{y \in f(D)} y \qquad (50)$$

For example, letting g be the identity function ($g(x) = x$),

$$\sum\nolimits_{I(10)} g(x) = \sum\nolimits_{k=1}^{10} k = 55 \qquad (52)$$

We next define a *field of domains* as a function that associates with every element of a domain a set of other domains. Generalizing equation (43), let A be a set of domains

$$A = \{D_1, D_2, \ldots\} \qquad (53)$$

and let $F: D \to S[A]$ be a function that maps domain elements to sets of domains. Then we define the field of domains $\mathcal{F}: D \to D \times S[A]$ associated with F as the set of ordered pairs

$$\mathcal{F}(F,D) = \{(d_i, F(D)) \mid d_i \in D\} \qquad (54)$$

and write

$$F(D) = \{f(d_i) \mid d_i \in D\} \qquad (55)$$

for the *range of the field of domains* corresponding to F.

Fields of domains are particularly useful because they can be defined dynamically. In a developmental simulation, for example, domains may be used to represent sets of cells. We will then often need to answer the following question: given any cell in an organism, what other cells interact with it? We can define a class of tissue domains T_i to represent any set of cells, up to and including the entire organism. Then the objects $t_{ij} \in T_i$ represent separate cells of the tissue domain T_i. Each t_{ij} has associated with it a set of other cells $\{t_{ik} \in T_i\}$ that are its neighbors. Let T_{ij} represent the neighbors of t_{ij}, i.e., $T_{ij} = \{t_{ik} \in T_i \mid t_{ik}\ \textit{is a neighbor of}\ t_{ij}\}$. T_{ij} is itself a tissue domain, and since T_{ij} is a set of cells that are all in T_i, we have $T_{ij} \in S[T_i]$. The mapping $N: t_i \in T_i \to T_{ij} \in S[T_i]$ gives us a *neighborhood function* that tells us which cells are neighbors of other cells. The corresponding field of domains $\mathcal{N}: T_i \to T_i \times S[T_i]$ formally defines this as the set of ordered pairs $\mathcal{N}(N,T_i) = \{(t_i, N(t_i)) \mid t_i \in T_i\}$, or more concisely, the range of $\mathcal{N}$ is expressed as $N(T_i) = \{N(t_i) \mid t_i \in T_i\}$. It is possible to implement the neighborhood field of domains via a potential function operator $V(t_i, t_j)$ that is nonzero only when there is some interaction between the two cells (we will give an example of one such potential function in the next section). We want $\mathcal{N}[t_i]$ to give us the ordered pair $(t_i, N[t_i])$, where

$$N[t_i] = \{t_j \in T \mid V(t_i, t_j) \ne 0\} \qquad (56)$$

We can now proceed to define dynamic operators. Let $g: \{\mathcal{E}[T_i]\} \to \mathcal{R}$ be a function. Then by generalizing our sum function (see equation 49), we can calculate a *sum over neighbors operator* $\sum_{\mathcal{N}} g$ as

$$\sum_{N[t_i]} g[t_j] = \Sigma(g, \mathfrak{E}[N[t_i]]) = \sum_{t_j \in N[t_i]} g(t_j) \qquad (57)$$

to give the sum over all neighbors of t_i of the function g. To illustrate the sum over neighbors, suppose that $\psi(t_i,t_j)$ is the electrostatic potential between cells t_i,t_j. The total electrostatic $\psi_T(t_i)$ potential at t_i is

$$\psi_T(t_i) = \sum_{N[t_i]} \psi(t_i,t_j) \qquad (58)$$

where $\mathcal{N}$ picks out those neighbors of t_i that are not electrically shielded from t_i.

Within Cellerator we define domains with uninstantiated Mathematica functions; the expand function is perform using up-values, e.g., the integer domain $I(m,n)$ is defined as

$$\texttt{expand[intDomain[m\_,n\_]] ^:= Range[m,n]} \qquad (59)$$

with similar definitions for the other domains. The notations of pure function, Apply, and Map very nicely fit into the concept of fields and operators on domains. However, this formal presentation gives us a mechanism whereby domains and fields could be implemented with any computer language, freeing us from the specifics of Mathematica.

2.3 The Organism-as-Graph Approach

In this approach a growing organism (or more likely, selected tissue within a growing organism) is represented by the following graph structure. A *graph* is composed of three things: a list of *nodes*, a list of *links*, and a *lineage tree*. Each node represents a particular cell; each link represents the interaction between two cells; and the lineage tree records the family tree of cell birth. The overall object hierarchy is illustrated in Figure 1.

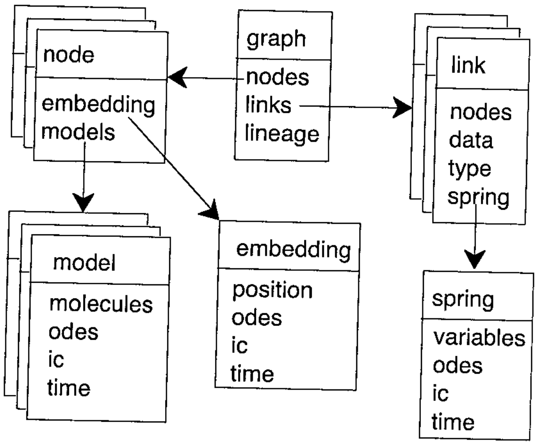


Figure 3. Structure of graph domains.

For example, a *graph domain* within Cellerator would be represented as

$$\begin{aligned}\texttt{g} = \texttt{graphDomain[nodes} &\rightarrow \{\texttt{n}_1,\texttt{n}_2,\ldots\},\\ \texttt{links} &\rightarrow \{\texttt{l}_1,\texttt{l}_2,\ldots\},\\ \texttt{lineage} &\rightarrow \texttt{T]}\end{aligned} \qquad (60)$$

Each node contains precisely one *embedding* and one or more *models*,

$$\begin{aligned}\texttt{n}_1 = \texttt{nodeDomain[embedding} &\rightarrow \texttt{e}_1,\\ \texttt{models} &\rightarrow \{\texttt{m}_1,\texttt{m}_2,\ldots\}\texttt{]}\end{aligned} \qquad (61)$$

The embedding describes the location of the cell in Cartesian coordinates, and an optional set of differential equations (and corresponding initial conditions) that describe the time evolution of those coordinates. In the simplest implementation only a single point is needed to describe a nodes embedding; however, there is no reason that additional information, such as the shape of a cell and its geometric relationship to other cells, could not be included.

$$\begin{aligned}&\texttt{e}_1 = \texttt{embeddingDomain[}\\ &\quad \texttt{position} \rightarrow \{\texttt{x}_1,\texttt{y}_1,\texttt{z}_1\},\\ &\quad \texttt{odes} \rightarrow \{\dot{\texttt{x}}_1 == \texttt{f}_1(\texttt{x}_1,\texttt{y}_1,\texttt{z}_1),\\ &\qquad\qquad \dot{\texttt{y}}_1 == \cdots\},\\ &\quad \texttt{ic} \rightarrow \{\texttt{x}_{10},\texttt{y}_{10},\ldots\},\\ &\quad \texttt{time} \rightarrow \texttt{t}_0\texttt{]}\end{aligned} \qquad (62)$$

The time field gives the time at which the initial conditions are applied.

Each node can contain one or more models. Each model domain contains a system of differential equations and associated parameters that describe some aspect of the cells signal transduction network. In theory, one could put all of the differential equations for a cell in a single model. In practice, however, it would be more convenient to have different models for distinct parts of the network. The differential equations in one model can refer to variables in another model (in fact, they can refer to variables in another node, as well). For example, to describe cell division it might be convenient to group the equations into separate models representing the G1 checkpoint, the G2 checkpoint, DNA replication, Mitosis, etc. For example, if we want the glycolytic system r illustrated in figure 2, all we need specify is

$$\begin{aligned}&\texttt{\{eqns,vars\}} = \texttt{interpret\{r\};}\\ &\texttt{m} = \texttt{modelDomain[}\\ &\quad \texttt{molecules} \rightarrow \texttt{vars,}\\ &\quad \texttt{odes} \rightarrow \texttt{eqns];}\end{aligned} \qquad (56)$$

Options that are not specified take on default values; for example, unspecified initial conditions and the initial time are set to zero. If we had a number of nodes $n_1,\ldots,n_k$, each of which requires a glycolytic system, we would just add indices to the variables, i.e., ADP becomes `ADP[i]`, i=1,..,k. This process is easily automated.

A minimum link contains pointers to two nodes. In Cellerator each node is numbered as it is added to the graph, so the pointers become integers representing the nodes. One can also add other information about the link in the data field. The link between nodes i and j is then represented as

$$\texttt{l}_1 = \texttt{linkDomain[nodes} \rightarrow \texttt{\{i,j\}]} \qquad (63)$$

Each link also contains an associated "spring" field that represents the dynamics of cell growth. These springs are used to define a potential function (Figure 4) that is used to optimize the position of the nodes after each growth step. The potential function for a node n_i is

$$V_{ij} = \frac{1}{2}\sum_j k_{ij}(|\mathbf{x}_i - \mathbf{x}_j| - d_{ij})^2 \tag{64}$$

where the sum is taken over all nodes n_j that are linked to n_i, $\mathbf{x}_i$ are the vector positions of the nodes, the k_{ij} are interaction strengths (zero for non-interacting nodes), and the d_{ij} give the desired distance between the nodes. The potential gets "turned off" when the interaction distance becomes too large in some sense; to prevent a discontinuity in the potential after cell division we modify (64) as:

$$V_{ij} = \frac{1}{2}\sum_j k_{ij} c_{ij}[(|\mathbf{x}_i - \mathbf{x}_j| - d_{ij})^2 - \mu] \tag{65}$$

where

$$c_{ij} = \begin{cases} 1, & d_{ij} \le d \\ 0, & d_{ij} > d \end{cases} \tag{66}$$

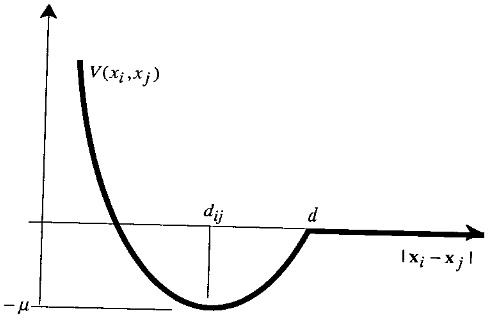


Figure 4. "Spring" potential (see equation 65).

The biological dynamics of growth are then described by assigning a relationship between d_{ij} and the model variables in nodes n_i and n_j. For example, suppose that the model stipulates that a cell's mass grows at a constant rate,

$$\frac{dM_i}{dt} = kM_i \;\; (M_i = M_{i0}e^{kt}) \tag{67}$$

and that we describe the cell as a sphere of radius R_i. Then if the density remains constant,

$$\frac{dR_i}{dt} = \frac{1}{3}kR_i \;\; (R_i = R_{i0}e^{kt/3}) \tag{68}$$

then we might constrain the equilibrium spring distance to be

$$d_{ij} = R_i + R_j \tag{69}$$

if the cells are assumed to be touching. After each step, the potential is then minimized (e.g., via gradient descent or simulated annealing) to determine the new position of the cell. For a small number of nodes the optimization can be replaced by adding a additional differential equations of the form

$$\frac{d\mathbf{x}_i}{dt} = -\nabla V_{ij} \tag{70}$$

This forces the solution to follow a gradient descent towards the minimum of the spring function. As a final note, the term "spring" here is used because of the form of the potential function, and does not actually describe position dynamics (otherwise, equation 70) would be second order in time. In fact, equation 70 would seem to give an exponential relaxation towards local minimum.

2.4 Variable Structure System

Biologically, the birth of new cells and the death of old cells occur as a result of the concentration of some chemical species passing a species. For example, the amount of ATP can fall so low that metabolic processes cease (death) or the quantity of S-phase promoting factor (SPF) may increase so high that DNA replication is induced (birth). In the first case, the number of active cells in the system ceases to exist; in the second case, an additional cell is added.

It is easier to represent death than birth. We can assign an indicator, or "aliveness," variable I_k to each cell k, where $I_k = 1$ if cell k is alive, and $I_k = 0$ if cell k dies. Then if we multiply the concentration of each chemical species x in cell k by I_k, all equations for a given cell effectively disappear when the cell dies. In principle we could do the same thing to describe birth. However, this would require that we know in advance the total maximum number of cells we would ever want to simulate. We would never be allowed to exceed this number. Besides being inelegant this could be a computationally exhaustive and extremely inefficient algorithm. Thus we propose an alternative mechanism to represent cell birth.

We assume that any change in the size of the system (i.e., the number of variables and differential equations and/or the values of corresponding initial conditions) is triggered by some threshold passage. Suppose that our system is composed of M+N variables, $\{x_i\}_{i=1,\ldots,N}$ and $\{w_i\}_{i=1,\ldots,M}$, where each of the N variables x_i can trigger some event when it passes a corresponding threshold T_i, and there are M remaining variables w_i in the total system. Then we define a set of flag variables $\{y_i, z_i\}_{i=1,\ldots,N}$, where there is one pair of flag variables for each trigger variable x_i. Then if the event is triggered the first time that $x_i > T_i$ we force the corresponding flag variables to satisfy the ODEs

$$\frac{dy_i}{dt} = \Theta(x_i - T_i), \;\; y_i(0) = 0 \tag{71}$$

$$\frac{dz_i}{dt} = \Theta(y_i), \;\; z_i(0) = 0 \tag{72}$$

where $\Theta(x)$ is the unit step function

$$\Theta(x) = \begin{cases} 0, & x < 0 \\ 1, & x \ge 0 \end{cases} \tag{73}$$

A similar equation can be written if the event is triggered by $x_i < T_i$. Then y_i increases linearly from zero whenever x_i is above threshold. Even if x_i falls back below threshold, y_i will remain positive; it stays fixed at whatever value it had when x_i falls back below threshold. The second new variable z_i increases linearly with time whenever y_i is positive. Thus z_i

measures the total amount of time that has elapsed since x_i first passed its threshold.

These new variables are then used as follows. We submit our entire system $\{x_i, y_i, z_i\}_{i=1,\ldots N} \cup \{w_i\}_{i=1,\ldots,M}$ to a fixed-structure solver for some time interval of length T. Here T is the total duration of numerical integration, and is not the integration step size, which is typically (several) orders of magnitude smaller. We would chose T to have a magnitude close to the expected time between triggering events (e.g., the length of the cell cycle). Presumably the solver will return the solutions to the differential equations (in the form of interpolatable functions of some sort) over the time interval $(0,T)$. Then we evaluate each of the variables z_i at the end of the solution interval (time $t=T$). If $z_i(T)=0 \quad \forall i=1,\ldots,N$ then no even occurred. Otherwise some event has occurred. Since it is possible that more than one of the z_i are nonzero we must examine them all to determine which one is the biologically meaningful one, i.e., which event occurred first. To determine this we calculate

$$z_k = \max_{i \le i \le N} z_i \tag{74}$$

Then x_k is the first variable that passed threshold, and this event occurred at time

$$t = T - z_k \tag{75}$$

So we accept the solution provided over the interval $(0, T-z_k)$ but reject the rest of the solution, over the interval $(T-z_k, T)$.. We define a new set of initial conditions at $T-z_k$ by evaluating the numerical solutions for x_i and w_i at that time, and setting $y_i(T-z_k)=0$ and $z_i(T-z_k)=0$. We then add whatever new variables are necessary to the system at $T-z_k$ (along with their corresponding differential equations and initial conditions). If some of these new variables can trigger events we must also add whatever new y_i and z_i variables are required to describe these new potential event triggers. Then we have a new system with a larger number of variables. The entire process is then repeated over the interval $(T-z_k,\ 2T-z_k)$ and is iterated as desired.

3. MIMIMAL DEVELOPMENTAL SYSTEM

The minimal developmental system can be illustrated using Goldbeter's minimal cascade for mitotic oscillations [3]. It should be noted that any system of differential equations can be used here, so long as there is at least one threshold that will trigger cell division. This particular system was chosen for illustrative purposes because of its elegant simplicity. The differential equations are

$$C' = v_i - \frac{v_d X C}{K_d + C} - k_d C \tag{76}$$

$$M' = V_1 \frac{1-M}{K_1 + (1-M)} - V_2 \frac{M}{K_2 + M} \tag{77}$$

$$X' = M V_{m3} \frac{1-X}{K_3 + (1-X)} - V_4 \frac{X}{K_4 + X} \tag{78}$$

where C, M, and X represent Cyclin concentration (C) and the fractions of active CDC2 kinase (M) and cyclin protease (X), respectively, with the additional constraint that

$$V_1 = \frac{V_{m1} C}{K_C + C} \tag{79}$$

All other parameters in the equations are constants.

The first differential equation (76) is straightforward; the first and third terms are creation and annihilation of C,

```
{∅⇌C,vi,kd}
```
(80)

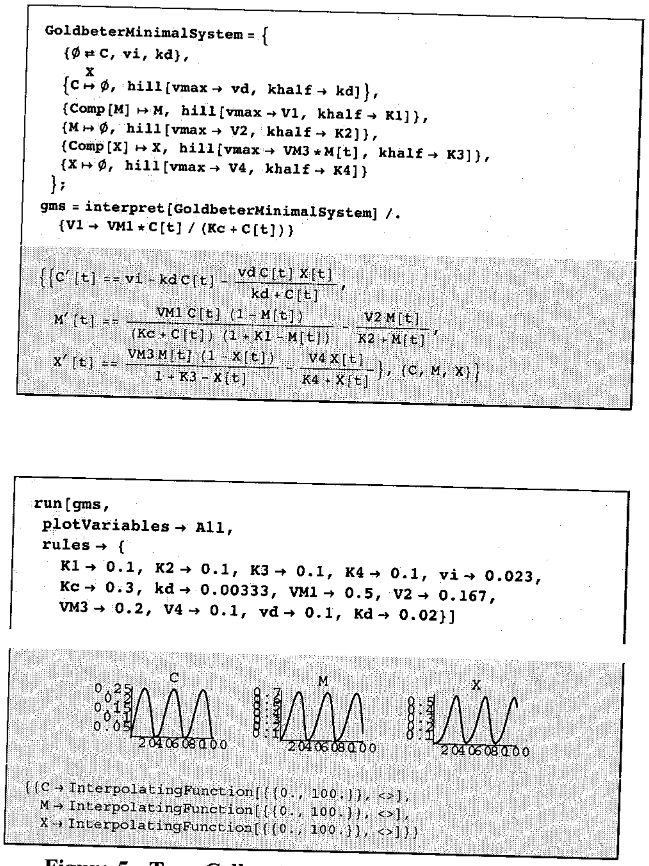
while the middle term is a hill-function annihilation of C catalyzed by X:

```
   X
{C↦∅,hill[vmax→vd,khalf→kd]}
```
(81)

Equations (77) and (78) use implicit mass conversation; for example, the first term in equations 77 is

```
{Comp[M]↦M,hill[vmax→V1,khalf→K1]}
```
(82)

and so forth for the remaining reactions. The constraint (79) is enforced utilizing Mathematica replacement rules. The entire set of reactions is illustrated in Figure 5.



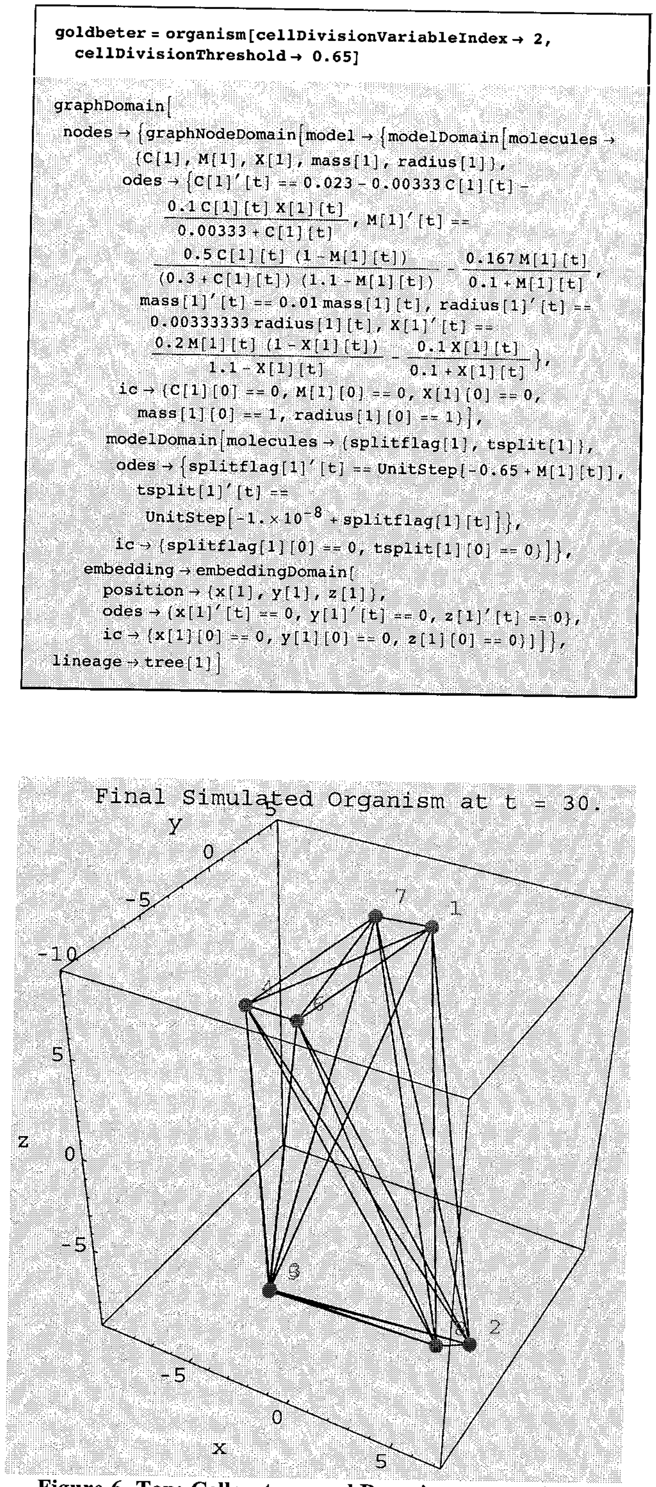
```
GoldbeterMinimalSystem = {
  {∅ ⇌ C, vi, kd},
    X
  {C ↦ ∅, hill[vmax → vd, khalf → kd]},
  {Comp[M] ↦ M, hill[vmax → V1, khalf → K1]},
  {M ↦ ∅, hill[vmax → V2, khalf → K2]},
  {Comp[X] ↦ X, hill[vmax → VM3 * M[t], khalf → K3]},
  {X ↦ ∅, hill[vmax → V4, khalf → K4]}
 };
gms = interpret[GoldbeterMinimalSystem] /.
  {V1 → VM1 * C[t] / (Kc + C[t])}
```

$$\left\{\left\{C'[t] == vi - kd\,C[t] - \frac{vd\,C[t]\,X[t]}{kd + C[t]},\ M'[t] == \frac{VM1\,C[t]\,(1 - M[t])}{(Kc + C[t])\,(1 + K1 - M[t])} - \frac{V2\,M[t]}{K2 + M[t]},\ X'[t] == \frac{VM3\,M[t]\,(1 - X[t])}{1 + K3 - X[t]} - \frac{V4\,X[t]}{K4 + X[t]}\right\}, \{C, M, X\}\right\}$$

```
run[gms,
 plotVariables → All,
 rules → {
   K1 → 0.1, K2 → 0.1, K3 → 0.1, K4 → 0.1, vi → 0.023,
   Kc → 0.3, kd → 0.00333, VM1 → 0.5, V2 → 0.167,
   VM3 → 0.2, V4 → 0.1, vd → 0.1, Kd → 0.02}]
```

```
{{C → InterpolatingFunction[{{0., 100.}}, <>],
  M → InterpolatingFunction[{{0., 100.}}, <>],
  X → InterpolatingFunction[{{0., 100.}}, <>]}}
```

Figure 5. Top: Cellerator arrows and corresponding differential equations generated by interpret for the mitotic oscillator (equations (76) to (79)). Bottom: The run function can solve the differential equations and plot results.



```
goldbeter = organism[cellDivisionVariableIndex → 2,
  cellDivisionThreshold → 0.65]

graphDomain[
 nodes → {graphNodeDomain[model → {modelDomain[molecules →
        {C[1], M[1], X[1], mass[1], radius[1]},
       odes → {C[1]′[t] == 0.023 - 0.00333 C[1][t] -
           (0.1 C[1][t] X[1][t])/(0.00333 + C[1][t]), M[1]′[t] ==
          (0.5 C[1][t] (1 - M[1][t]))/((0.3 + C[1][t]) (1.1 - M[1][t])) - (0.167 M[1][t])/(0.1 + M[1][t]),
        mass[1]′[t] == 0.01 mass[1][t], radius[1]′[t] ==
         0.00333333 radius[1][t], X[1]′[t] ==
         (0.2 M[1][t] (1 - X[1][t]))/(1.1 - X[1][t]) - (0.1 X[1][t])/(0.1 + X[1][t])},
      ic → {C[1][0] == 0, M[1][0] == 0, X[1][0] == 0,
        mass[1][0] == 1, radius[1][0] == 1}],
     modelDomain[molecules → {splitflag[1], tsplit[1]},
      odes → {splitflag[1]′[t] == UnitStep[-0.65 + M[1][t]],
        tsplit[1]′[t] ==
         UnitStep[-1. × 10^-8 + splitflag[1][t]]},
      ic → {splitflag[1][0] == 0, tsplit[1][0] == 0}]},
   embedding → embeddingDomain[
     position → {x[1], y[1], z[1]},
     odes → {x[1]′[t] == 0, y[1]′[t] == 0, z[1]′[t] == 0},
     ic → {x[1][0] == 0, y[1][0] == 0, z[1][0] == 0}]]},
 lineage → tree[1]]
```

Figure 6. Top: Cellerator graphDomain representing a single celled organism with a minimal developmental system modelDomain; Bottom: Illustration of organism-as-graph structure after 8 generations.

The Cellerator run function will automatically perform a simulation (numerically solve the ODEs using NDSolve) and plot the resulting concentration profiles, as illustrated in the bottom of Figure 5. It is not necessary to interpret the reactions first; this is done in Figure 5 merely for illustrative purposes. To perform a simulation a graphDomain with a single node and a model given by the minimal system is first created as the initial one-celled organism (see Figure 6). An index indicating the cell number is added to each variable in the system.

The second modelDomain in figure 6, implements the variable-structure threshold described in section 2.4, with variables splitflag and tsplit represnting y_1 and z_1 respectively. The system can then be repeatedly integrated using NDSolve, testing the values of the flags after each step. Whenever a threshold is passed, a new node is added to the graph. The concentrations of *C*, *M*, and *X* are randomly split between parent and child cell after each cell division so that the total number of molecules remains fixed before and after cell division. Figure 8 shows a snapshot of the graph when the organism has grown to eight cells. Each time a cell divides, a new cell number (equal to *N*+1 where *N* is the number of cells in the graph just before cell division) is assigned to the child cell while the parent retains its old cell number (the first cell is number 1). the corresponding node on the lineage tree is replace by a binary subtree tree[parent,child]. The final lineage tree is

```
tree[tree[tree[1, 7], tree[4, 6]],
      tree[tree[2, 8], tree[3, 5]]]        (83)
```

A different simulation would produce a differently shaped organism because of the nature of the random number assignments.

4. DISCUSSION

We have presented a graph-based methodological paradigm for computational simulations in developmental biology. Multi-cellular systems (organisms) are represented as graphs whose nodes and links represent cells and intercellular interactions, respectively. This approach has a natural hierarchical generalization that makes it very amenable to multi-scale analyses: when more detail is needed, the data stored in a graph node can be expanded into a graph representing the intracellular signal transduction network (STN) of that cell. Nodes on the STN are themselves progressively more detailed signal transduction sub-networks, and so forth, down to a single molecular species, if so desired. Because it would be pointless to try to deterministically simulate every chemical species in every cell at every time such a multi-scale approach becomes essential.

The concept of a canonical form is central to the Cellerator paradigm. Although we have introduced a new terminology to describe this process, the postulate that such forms exist underlies any object-oriented methodology. At each level of information processing, there are both input and output canonical forms. The output forms can them be fed back into the system as input forms for the next stage of processing. How these are implemented are platform and language dependent. It is the canonical forms themselves that are essential to the paradigm, not their implementation. One can thus visualize a succession of canonical forms (Table 3). The various levels indicated in Table 3 are for illustrated purposes only; the actual names and succession of levels is not a core element of the paradigm. Central to this visualization is user-interaction at any level: the ability to modify not only the initial cartoon figure but also the Cellerator arrows, the biochemical equations, the DAEs (differential/algebraic equations), etc. We have described canonical forms for the notebook interface and the translator in detail, and have illustrated how they may be packaged into graphs for developmental simulations.

We have also shown how the simulation kernel can implement a variable structure system using a pre-packaged fixed structure solver. At the present time, the Cellerator computer program has a Mathematica notebook interface; we plan to add a graphical user interface in the future. There is also no reason why the core translator could not be used with other front ends as well.

Table 3. Canonical forms for different levels of the Cellerator paradigm. STN: Signal Transduction Network; DAE: Differential-Algebraic Equation; ODE: Ordinary Differential Equation.

Level	Input Form	Output Form
Web Interface	Web Database Record	Graph-Representation of STN
User Interface	Cartoon Figure	Graph representation of STN
Notebook Interface	Cellerator Arrow	Chemical Equations
Translator	Chemical Equations	DAEs, ODEs
Solver	DAEs, ODEs	Numerical Solutions
Simulation Kernel	Numerical Solutions plus Graph	Modified Graph

In the past it has been necessary to manually translate chemical networks into differential equations and then solve them numerically. Bhalla [1] has noted the need to systematically study interacting pathways with a standardized scheme, and has described several networks with mass-action kinetics using the Genesis [2] simulator. A few other authors have also proposed or implemented various forms of arrow-based notations for biochemical simulations. Kohn [7] has proposed an arrow-based system for illustrating complex signal transduction networks. A network-to-graph representation translation of such a design should be straightforward to implement. Ichikawa [6] has developed palette-based graphical user interfaces for describing small biochemical systems and translating them into differential equations that can be subsequently solved with other utilities.

In addition there is a growing collection of tools that under development or that have already been released that can (or will be able to) simulate signal transduction or metabolic networks in a variety of specialized domains (*e.g.,* BioSpice, DBSolve, E-Cell, Genesis, Gepasi, M-Cell, Neuron, Stochsim, V-Cell, XSIM; see [5] for references). A standardized, freely accessible web-based database (such as KEGG) including all necessary simulation parameters and associated information that can be directly read into a user's chosen biology simulation workbench is still missing from the picture.

Common use of a standardized data transfer protocol such as the systems biology markup language (SBML) proposed by Hucka et al [6]. would be a fundamental step towards allowing these specialized tools to freely interface with one another.

5. ACKNOWLEDGMENTS

This work was supported in part by the Whittier Foundation and the ERATO Kitano Symbiotic Systems Project. We would like to express our thanks for valuable discussions with A. Levchenko, E. Meyerowitz, B. Wold, H. Boulori and the members of the Caltech Systems Biology Workbench Development Group.

6. REFERENCES

[1] Bhalla, U.S., Iyengar, R. Emergent properties of networks of biological signaling pathways. Science 283:381-387.

[2] Bower, J.M., Beeman.D. The book of Genesis. Spirnger Verlag, Berlin (1998).

[3] Goldbeter, A. A minimal cascade model for the mitotic oscillator involving cyclin and cdc2 kinase. Proc. Natl. Acad. Sci. USA 88:9107-1101 (1991).

[4] Goldbeter,A., Lefever, R. Dissipative structures for an allosteric model: application to glycolytic oscillations. Biophys J. 12:1302-1315 (1972).

[5] Hucka M., Finney, A., Sauro, H., Boulori, H. Systems biology markup language (SMBL) level 1: Structures and facilities for basic model definitions. 2 March 2001. http://www.cds.Caltech.edu/erato.

[6] Ichikawa, K. A-Cell: graphical user interface for the construction of biochemical reaction models. Bioinformatics, 17:483-484 (2001).

[7] Kohn, K.W. Molecular interaction map of the mammalian cell cycle control and DNA repair systems. Mol. Biol. Cell. 10:2703-2734 (1999).

[8] Mjolsness, E. Trainable gene regulation networks with applications to Drosophila pattern formation. In: Computational Models of Genetic and Biochemical Networks, ed. Bower,J.M., Bolouri,H. MIT Press (2000).

[9] Monod, J., Wyman, J., Changeux, J.-P. On the nature of allosteric transitions: A plausible model. J. Mol. Biol. 12: 88-118 (1965).

[10] Shapiro, B.E, Levchenko, A. , Mjolsness, E. Automatic model generation for signal transduction with applications to MAP-Kinase pathways. In: Foundations of Systems Biology, ed. H. Kitano. MIT Press, Cambridge, MA (2001).

[11] Wolfram, S. The Mathematica Book. Fourth Edition. Cambridge University Press, New York (1999).